

DEVELOPMENT OF MOTOR CONTROL USING
GRAPHICAL USER INTERFACE

KHAIRUL ANUAR BIN ARIS

UNIVERSITY MALAYSIA PAHANG

ABSTRACT

DC Motor control is very common in robotic application. The developments of this kind of project are widely used in most electronic devices nowadays. There are many application that have been developed based on motor control in electronic field such as in automation, Flexible Manufacturing System (FMS) and Computer Integrated Manufacturing (CIM). The purpose of this project is to develop the Graphical User Interface of Motor Control through MATLAB GUIDE, interface the MATLAB GUI with hardware via communication port and control the DC motor through MATLAB GUI. By using MATLAB GUIDE, it provides a set of tools which simplify the process of laying out and programming GUIs and interface with PIC via serial communication port to control the DC motor. The PIC is used to control motor. As a result, the DC motor is able to be controlled through MATLAB GUI and interface the MATLAB GUI with PIC via serial communication port.

CHAPTER 1

INTRODUCTION

1.1 Overview

The serial port found on the back of the most PC and it is extremely useful for robotics work. Variety devices are configured to communicate via a serial port.

This Project is focus on designing the Graphical User Interface (GUI) through MATLAB to control the DC motor using PIC. The PIC is a programmable interface devices or controller between PC (MATLAB GUI) and the DC motor. The main contribution of this project is the interfacing of the MATLAB with PIC and Graphical User Interface (GUI).

The Peripheral Interface Controller (PIC) use in this project is as controller device between Personal Computer and the DC motor to control DC motor. The PIC is use because of wide availability and economical. Beside that PIC is a free development tools and can perform many function without needed extra circuitry. The PIC is program using the PICBasic Pro Compiler. The PicBasic Pro Compiler produces code that may be programmed into a wide variety of PICmicro

microcontrollers having from 8 to 84 pins and various on-chip features including A/D converters, hardware timers and serial ports. The purpose using MATLAB in creating the GUI is because it already has Graphical User Interface Development Environment (GUIDE) that provides a set of tools for creating GUI. These tools simplify the process of laying out and programming GUIs.

The GUI create in MATLAB with appropriate coding will control the DC motor via serial port that interface with the PIC. There are many advantage by using the DC motor, among that the DC motor has no adverse effect on power quality and the speed is proportional to the magnetic flux.

1.2 Objective

At the end of this Project:

- i. Able to control DC motor through MATLAB GUI.
- ii. Able to interface the MATLAB GUI with hardware using PIC.

The important part of this project is to interface the MATLAB GUI with the PIC. This part is done if the PIC produces a signal. The output from PIC will monitor by using the oscilloscope. After that the DC motor can be control via MATLAB GUI.

1.3 Scope of Project

The scopes of this project are laying out the GUI in MATLAB GUIDE and create programming for the GUI's. Secondly Prepare the PIC circuitry and serial

connection (DB9) circuit for interfacing part. And the third part is creating program for PIC using PICBasic Pro Compiler to control the DC motor.

1.4 Problem Statement

The main objective in this project to interface the MATLAB GUI with the PIC. It is a difficult part to develop the program for MATLAB and the PIC simultaneously to make the interfacing part. By using the PicBasic Pro Compiler software to develop programming to control DC motor, it can reduces the difficulty by comprises a list of statements that written in a programming language like assembler, C, or BASIC. With this opportunity, the men in charge do not have to take long time to written and troubleshoot the program.

1.5 Thesis Organization

This thesis consists of five chapters including this chapter. The contents of each chapter are outlined as follows;

Chapter 2 contains a detailed description each part of project. It will explain about the MATLAB GUIDE, PIC, and DC motor. Chapter 3 includes the project methodology. This will explain how the project is organized and the flow of the process in completing this project. Chapter 4 presents the expected result of simulation runs using MATLAB GUIDE. Finally the conclusions for this project are presented in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Graphical User Interface (GUI)

2.1.1 General Definition of GUI

A **graphical user interface** (or **GUI**, often pronounced "gooey"), is a particular case of user interface for interacting with a computer which employs graphical images and widgets in addition to text to represent the information and actions available to the user [4][5]. Usually the actions are performed through direct manipulation of the graphical elements.

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive [4].

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth [2][4]. A true GUI includes standard formats for representing text and graphics [4]. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on a pushbutton, the GUI should initiate the action described on the label of the button.

Many DOS programs include some features of GUIs, such as menus, but are not *graphics based*. Such interfaces are sometimes called *graphical character-based user interfaces* to distinguish them from true GUIs [4].

2.1.2 MATLAB GUI

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand [1] [2] [16].

The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders — just to name a few. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components [1] [2] [3].

2.1.3 Operation of GUI

Each component, and the GUI itself, is associated with one or more user-written routines known as callbacks. The execution of each callback is triggered by a particular user action such as, mouse click, pushbuttons, toggle buttons, lists, menus, text boxes, selection of a menu item, or the cursor passing over a component and so forth [1] [2].

Clicking the button triggers the execution of a callback [1]. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back [1] [2].

This kind of programming is often referred to as event-driven programming. The event in the example is a button click. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB GUIs, these events usually take the form of user interactions with the GUI. The writer of a callback has no control over the sequence of events that leads to its execution or, when the callback does execute, what other callbacks might be running simultaneously [1].

Callbacks

- Routine that executes whenever you activate the uicontrol object
- Define this routine as a string that is a valid MATLAB expression or the name of an M-file
- The expression executes in the MATLAB workspace.

2.1.4 A brief introduction of GUIDE

GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs [1].

- GUIDE is primarily a set of layout tools
- GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI
 - This M-file also provides a framework for the implementation of the callbacks - the functions that execute when users activate a component in the GUI [1].

2.1.4.1 Two Basic Task in Process of implementing a GUI

The two basic tasks in Process of implementing a GUI is first, laying out a GUI where MATLAB implement GUIs as figure windows containing various styles of uicontrol (User Interface) objects. The second task is programming the GUI, where each object must be program to perform the intended action when activated by the user of GUI [14].

2.2 DC Motors

2.2.1 Introduction

Electric motors are everywhere! In a house, almost every mechanical movement that you see around you is caused by a DC (direct current) electric motor. An electric motor is a device that transforms electrical energy into mechanical energy by using the motor effect [7] [8].

Every DC motor has six basic parts -- axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors, the external magnetic field is produced by high-strength permanent magnets. The stator is the stationary part of the motor -- this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator [7] [8].

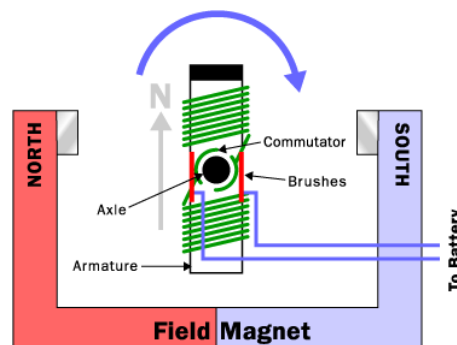


Figure 2.1: Part of an Electric Motor

Industrial applications use dc motors because the speed-torque relationship can be varied to almost any useful form -- for both dc motor and regeneration applications in either direction of rotation. Continuous operation of dc motors is

commonly available over a speed range of 8:1. Infinite range (smooth control down to zero speed) for short durations or reduced load is also common [6].

Dc motors are often applied where they momentarily deliver three or more times their rated torque. In emergency situations, dc motors can supply over five times rated torque without stalling (power supply permitting) [6].

Dc motors feature a speed, which can be controlled smoothly down to zero, immediately followed by acceleration in the opposite direction -- without power circuit switching. And dc motors respond quickly to changes in control signals due to the dc motor's high ratio of torque to inertia [6] [7].

2.2.2 The Advantages

The greatest advantage of DC motors may be speed control. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage and/or the field current will change the rotor speed [7].

- Today, adjustable frequency drives can provide precise speed control for AC motors, but they do so at the expense of power quality, as the solid-state switching devices in the drives produce a rich harmonic spectrum. The DC motor has no adverse effects on power quality [6] [7].

2.2.3 The drawbacks

- Power supply, initial cost, and maintenance requirements are the negatives associated with DC motors
- Rectification must be provided for any DC motors supplied from the grid. It can also cause power quality problems.
- The construction of a DC motor is considerably more complicated and expensive than that of an AC motor, primarily due to the commutator, brushes, and armature windings. An induction motor requires no commutator or brushes, and most use cast squirrel-cage rotor bars instead of true windings — two huge simplifications [6].

2.2.4 Type of DC Motor

2.2.4.1 Stepper motors

A **stepper motor** is a brushless, synchronous electric motor that can divide a full rotation into a large number of steps, for example, 200 steps. Thus the motor can be turned to a precise angle [7]. A stepper motor is an electromechanical device which converts electrical pulses into discrete mechanical movements and is a unique type of dc motor that rotates in fixed steps of a certain number of degrees. Step size can range from 0.9 to 90 degree [6] [7].

The speed of the motor shafts rotation is directly related to the frequency of the input pulses and the length of rotation is directly related to the number of input pulses applied. The motors rotation has several direct relationships to these applied input pulses. The sequence of the applied pulses is directly related to the direction of

Motor shafts rotation [6] [8]. The stepper motors has an excellent response to start-up, stopping and reverse [7].

There are three main of stepper motor type. First is Permanent Magnet (PM) Motors second is Variable Reluctance (VR) Motors and the third is Hybrid Motors.

2.2.4.2 Brushless DC motors

- A **brushless DC motor (BLDC)** is an AC synchronous electric motor that from a modeling perspective looks very similar to a DC motor.
- In a BLDC motor, the electromagnets do not move; instead, the permanent magnets rotate and the armature remains static.
- In order to do this, the brush-system/commutator assembly is replaced by an
- Intelligent electronic controller. The controller performs the same power-distribution found in a brushed DC-motor, but using a solid-state circuit rather than a commutator/brush system [6].

2.2.4.3 Coreless DC motors

- Optimized for rapid acceleration, these motors have a rotor that is constructed without any iron core.
- Because the rotor is much lighter in weight (mass) than a conventional rotor formed from copper windings on steel laminations, the rotor can accelerate much more rapidly, often achieving a mechanical time constant under 1 ms.

- These motors were commonly used to drive the capstan(s) of magnetic tape drives and are still widely used in high-performance servo-controlled systems [6].

2.3 PIC Microcontroller

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650 originally developed by General Instrument's Microelectronics Division[9] [10].

PICs are popular with developers due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability[9].

2.3.1 ORIGINS

1. The original PIC was built to be used with GI's new 16-bit CPU, the CP1600. While generally a good CPU, the CP1600 had poor I/O performance, and the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU.
2. The PIC used simple microcode stored in ROM to perform its tasks, and although the term wasn't used at the time, it is a RISC design that runs one instruction per cycle (4 oscillator cycles).

3. In 1985 General Instruments spun off their microelectronics division, and the new ownership cancelled almost everything — which by this time was mostly out-of-date. The PIC, however, was upgraded with EPROM to produce a programmable channel controller, and today a huge variety of PICs are available with various on-board peripherals (serial communication modules, UARTs, motor control kernels, etc.) and program memory from 512 words to 32k words and more[9].

2.3.2 PIC Microcontroller Option

A PIC Microcontroller chip combines the function of microprocessor, ROM program memory, some RAM memory and input-output interface in one single package which is economical and easy to use [10][14].

The PIC – Logicator system is designed to be used to program a range of 8, 18, 28 pin reprogrammable PIC microcontroller which provide a variety of input – output, digital input and analogue input options to suit students project uses [10].

Reprogrammable “FLASH Memory” chips have been selected as the most economical for student use. If a student needs to amend to control system as the project is evaluated and developed, the chip can simply be taken out of the product and reprogrammed with an edited version of the flow sheet [10].

The PIC devices generally feature is sleep mode (power saving), watchdog timer and various crystal or RC oscillator configuration, or an external clock.

2.3.3 Variants

Within a series, there are still many device variants depending on what hardware resources the chip features [9].

- general purpose i/o pins
- internal clock oscillators
- 8/16 Bit Timers
- Internal EEPROM Memory
- Synchronous/Asynchronous Serial Interface USART
- MSSP Peripheral for I²C and SPI Communications
- Capture/Compare and PWM modules
- Analog-to-digital converters
- USB, Ethernet, CAN interfacing support
- external memory interface
- Integrated analog RF front ends (PIC16F639, and rfPIC)
- KEELOQ Rolling code encryption peripheral (encode/decode)

2.4 PIC Basic Pro Compiler

The PicBasic Pro Compiler (or PBP) makes it even quicker and easier to program Microchip Technology's powerful PICmicro microcontrollers (MCUs). The English-like BASIC language is much easier to read and write than the quirky Microchip assembly language [11].

The PicBasic Pro Compiler is "BASIC Stamp II like" and has most of the libraries and functions of both the BASIC Stamp I and II. Being a true compiler, programs execute much faster and may be longer than their Stamp equivalents. PBP is not quite as compatible with the BASIC Stamps as our original [11] [17].

The PicBasic Pro Compiler produces code that may be programmed into a wide variety of PICmicro microcontrollers having from 8 to 84 pins and various on-chip features including A/D converters, hardware timers and serial ports [11].

2.5 LDmicro

LDmicro generates native code for certain Microchip PIC16 and Atmel AVR microcontrollers. Usually software for these microcontrollers is written in a programming language like assembler, C, or BASIC. A program in one of these languages comprises a list of statements. These languages are powerful and well-suited to the architecture of the processor, which internally executes a list of instructions. PLCs, on the other hand, are often programmed in 'ladder logic.'

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the methodology of this project. It describes on how the project is organized and the flow of the steps in order to complete this project. The methodology is diverged in two parts, which is developing the hardware to interface with MATLAB. The other is developing the programming for MATLAB and the PIC to control DC motor.

3.2 Methodology

There are three mains method in order to develop this project. Before the project is developing using MATLAB, it is needed to do the study on MATLAB GUIDE and the hardware (especially PIC). The flowchart in Figure 3.1 illustrated the

sequence of steps for this project. The first method is developing GUI in MATLAB and programs every GUI component. Secondly is to develop PIC programming to control 5V DC and stepper motor. And lastly is hardware design which is use to interface with MATLAB GUI.

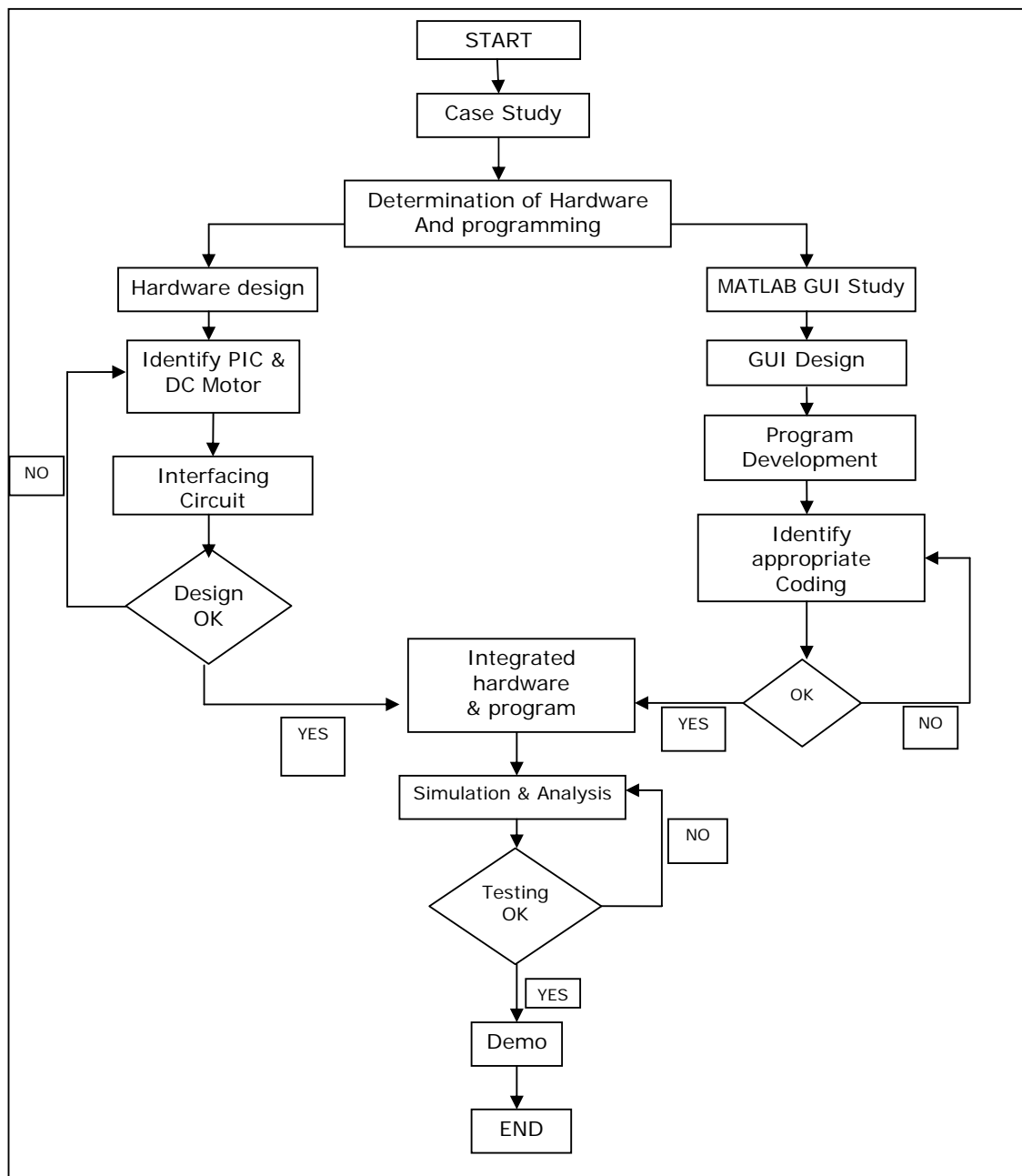


Figure 3.1: Flowchart for Whole Project

Figure 3.1 show the flow of the whole project. The project begins after registering the PSM title with doing case study about the project. The flow of the project is separate into two main tasks that are hardware design and MATLAB GUI design. In hardware design part flow, the main target is to create appropriate programming for PIC to interface with personal computer via serial port to control DC motor. The second part, the prior task is to develop program in MATLAB to interface with PIC and the DC motor. After that the both part is combine and do the analysis until achieve the needed objective. The main contribution of this project is to interface MATLAB GUI with the PIC.

3.2.1 Development MATLAB GUI Using MATLAB GUIDE

GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.

There are 5 steps in build the MATLAB GUI. First Use a MATLAB tool called guide (GUI Development Environment) to layout the components that show in figure 3.2. This tool allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in it. The basic component of the MATLAB GUI is shown in Table 3.1.

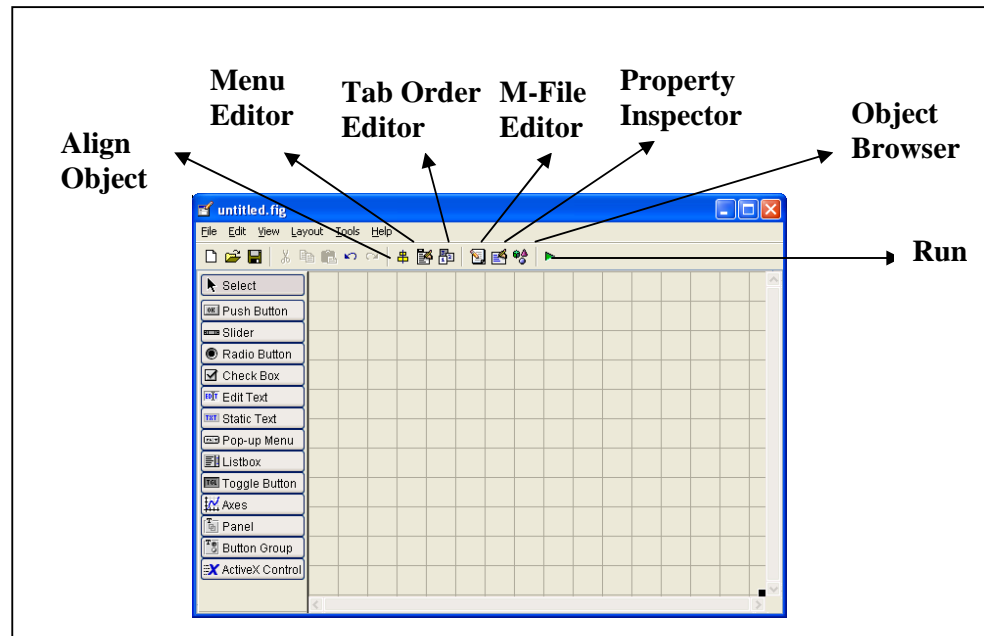


Figure 3.2: MATLAB GUIDE Layouts

Next is Use a MATLAB tool called the Property Inspector (built into guide) to give each component a name (a "tag") and to set the characteristics of each component, such as its color, the text it displays, and so on. After that, save the figure to a file. When the figure is saved, two files will be created on disk with the same name but different extents. The fig file contains the actual GUI that has been created, and the M-file contains the code to load the figure and skeleton call backs for each GUI element. These two files usually reside in the same directory. They correspond to the tasks of laying out and programming the GUI. When you lay out the GUI in the Layout Editor, your work is stored in the FIG-file. When you program the GUI, your work is stored in the corresponding M-file.

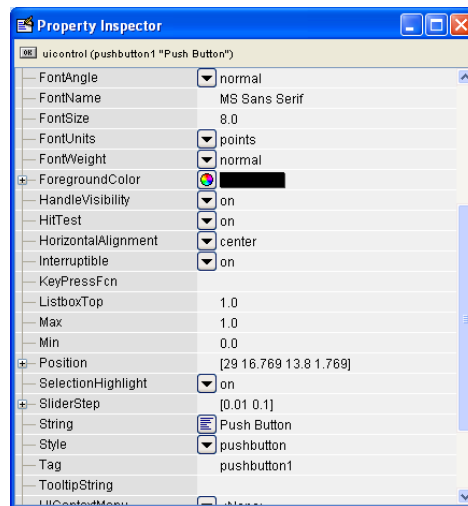


Figure 3.3: Property Inspector

Table 3.1: Basic MATLAB GUI Component [12]

Element	Created By	Description
Graphical Controls		
Pushbutton	uibutton	A graphical component that implements a pushbutton. It triggers a callback when clicked with a mouse.
Toggle button	uibutton	A graphical component that implements a toggle button. A toggle button is either "on" or "off," and it changes state each time that it is clicked. Each mouse button click also triggers a callback.
Radio button	uibutton	A radio button is a type of toggle button that appears as a small circle with a dot in the middle when it is "on." Groups of radio buttons are used to implement mutually exclusive choices. Each mouse click on a radio button triggers a callback.
Check box	uibutton	A check box is a type of toggle button that appears as a small square with a check mark in it when it is "on." Each mouse click on a check box triggers a callback.
Edit box	uibutton	An edit box displays a text string and allows the user to modify the information displayed. A callback is triggered when the user presses the Enter key.
List box	uibutton	A list box is a graphical control that displays a series of text strings. A user can select one of the text strings by single- or double-clicking on it. A callback is triggered when the user selects a string.
Popup menus	uibutton	A popup menu is a graphical control that displays a series of text strings in response to a mouse click. When the popup menu is not clicked on, only the currently selected string is visible.
Slider	uibutton	A slider is a graphical control to adjust a value in a smooth, continuous fashion by dragging the control with a mouse. Each slider change triggers a callback.
Static Elements		
Frame	uibutton	Creates a frame, which is a rectangular box within a figure. Frames are used to group sets of controls together. Frames never trigger callbacks.
Text field	uibutton	Creates a label, which is a text string located at a point on the figure. Text fields never trigger callbacks.
Menus and Axes		
Menu items	uimenu	Creates a menu item. Menu items trigger a callback when a mouse button is released over them.
Context menus	uicontextmenu	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.
Axes	axes	Creates a new set of axes to display data on. Axes never trigger callbacks.

After laying out the GUI component and set the property, the GUI will be look like in figure 3.4 for example according to the user creativity.

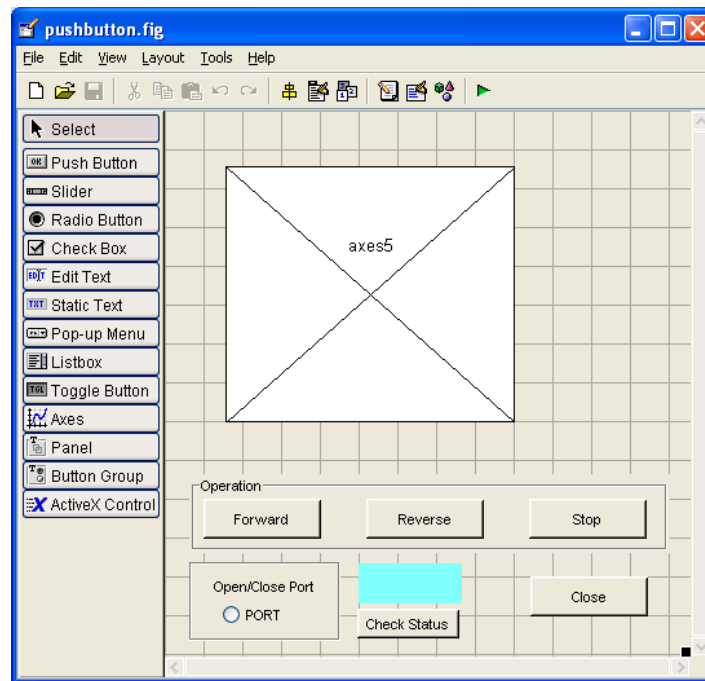


Figure 3.4: Example GUI

And finally write code to implement the behavior associated with each callback function in m-files show in figure 3.5. A callback is a function that writes and associates with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming. This last step is the difficult one and has to make an extra reading on how to write the coding before the GUI component can perform some task that user desire.

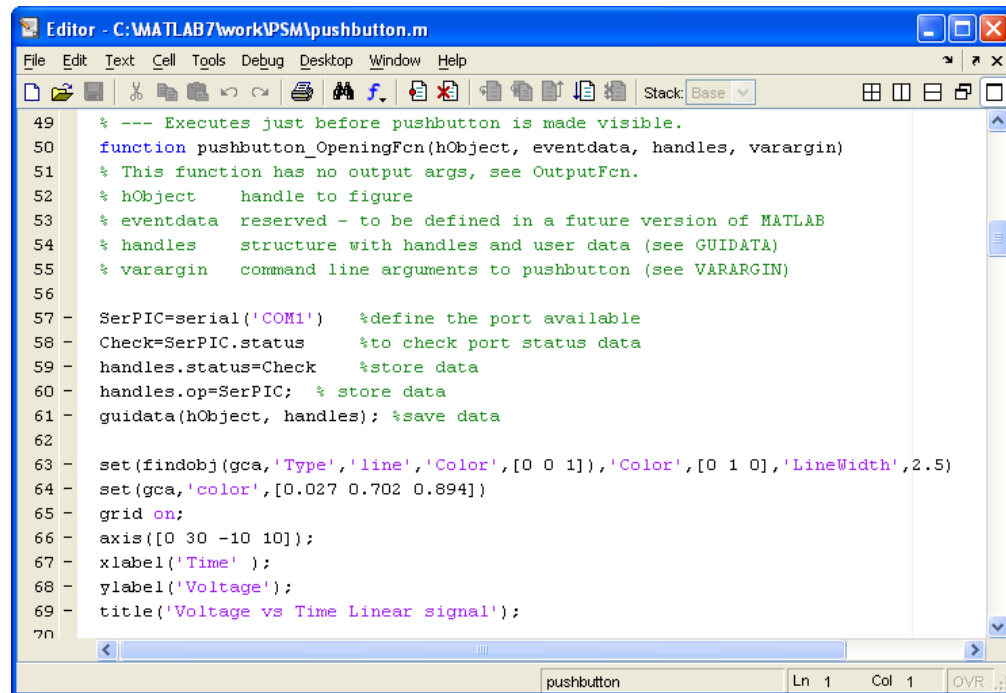


Figure 3.5: Example M-files for GUI

3.2.2 Build MATLAB Programming

After laying out the GUI, it needs to be programmed. The code is to write controls how the GUI responds to events such as button clicks, slider movement, menu item selection, or the creation and deletion of components. This programming takes the form of a set of functions, called callbacks, for each component and for the GUI figure itself.

A callback is a function that writes and associates with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming.

The GUI figure and each type of component have specific kinds of callbacks with which it can be associated. The callbacks that are available for each component are defined as properties of that component. Each kind of callback has a triggering mechanism or event that causes it to be called. The kind of callback is shown in table 3.2.

Table 3.2: Kind of Callback

Callback Property	Triggering Event	Components
DeleteFcn	Component deletion. It can be used to perform cleanup operations just before the component or figure is destroyed.	Axes, figure, button group, context menu, menu, panel, user interface controls
KeyPressFcn	Executes when the user presses a keyboard key and the callback's component or figure has focus.	Figure, user interface controls
ResizeFcn SelectionChangeFcn	Executes when a user resizes a panel, button group, or figure whose figure. Resize property is set to On. Executes when a user selects a different radio button or toggle button in a button group component.	Button group, figure, panel Button group
WindowButtonDownFcn	Executes when you press a mouse button while the pointer is in the figure window.	Figure

WindowButtonMotionFcn	Executes when you move the pointer within the figure window.	Figure
WindowButtonUpFcn	Executes when you release a mouse button.	Figure
ButtonDownFcn	Executes when the user presses a mouse button while the pointer is on or within five pixels of a component or figure. If the component is a user interface control, its Enable property must be on.	Axes, figure, button group, panel, user interface controls
Callback	Component action. Executes, for example, when a user clicks a push button or selects a menu item.	Context menu, menu, user interface controls
CloseRequestFcn	Executes before the figure closes.	Figure
CreateFcn	Component creation. It can be use to initialize the component when it is created. It executes after the component or figure is created, but before it is displayed.	Axes, figure, button group, context menu, menu, panel, user interface controls